GlobalLogic®

Real-time Premium Calculation Using IoMT in Health Insurance

by Ashutosh Gupta,
Senior Solution Architect, Global Logic

Introduction

When we buy health insurance, the health insurance industry uses traditional approaches to calculate premiums. The current premium calculation method considers only a few factors to calculate the premium instead of using a holistic view. These factors are dynamic because it involves humans. The health insurance industry considers factors based on the insured's historical data, including age, smoking, diabetes, hypertension, etc., to calculate health insurance premiums.

With the advancement of technology, other factors may impact the health of a person. For example, not all non-smokers in the age 35-40 range have similar health. These omissions can lead to losses or low profit for the insurance company.

These general categories can have a negative effect on the insured as well. A person who makes a healthy lifestyle a top priority pays the same amount of premium as a person in the same age group who lives an unhealthy lifestyle.

A better option could calculate premiums based on the insured's current lifestyle, rather than simply using traditional factors. Internet of Medical Things (IoMT) wearable gizmos such as smartwatches, wearable pedometers, and similar, and apps such as Google Fit could make it possible to calculate the "heart points" to determine how healthy a person is, in conjunction with traditional metrics. Modern IoMT devices can also track and record heart rate, temperature, and respiratory rate with built-in sensors.

Modern technology can help create a system for premium calculation, which can be lucrative for both the insurance company and the insured. The insurance company could charge higher-risk individuals with an unhealthy lifestyle a higher premium while giving healthy insureds a discount on premium since their claims risk would be lower.

This paper focuses on how IoMT devices can be used in real-time premium calculation, creating a win-win situation.

Using IoMT Devices for Premium Calculation

The solution would include generating data from IoMT devices, collected, and converted to a format that allows it to be analysed. The collected data would be provided to the rating engine and, combined with traditional metrics to calculate the premium.

Design Considerations

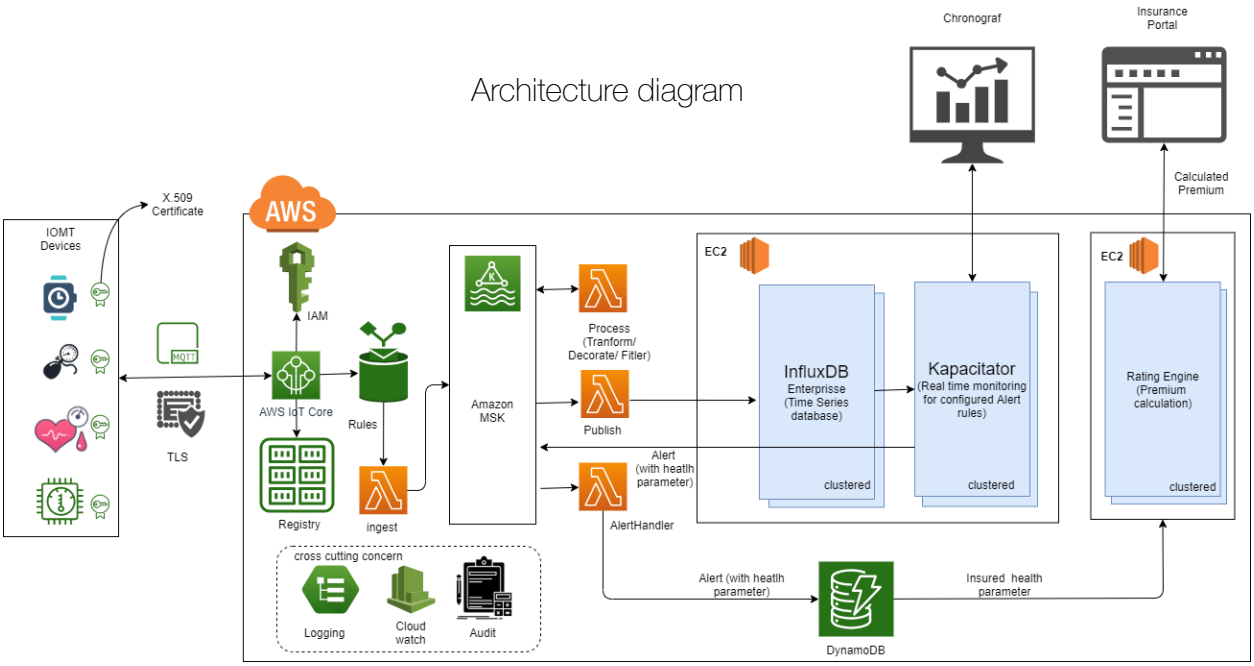
Several design considerations would need to be in place to take advantage of the big data that IoMT devices could generate.

- Scalable by design: There is the potential for millions of devices to be connected to the system, so any design solution must be scalable without requiring any changes to the foundational architecture.
- Modular Design Approach: Any system should be designed with modularity and flexibility in mind so that any module can be deployed and scaled independently.
- Stateless Design and Data Persistence: All services and components should be stateless so that the request from the same device need not stick to a specific component.
- Maximum Usage of SaaS/PaaS: Leverage Software-as-a-Service(SaaS)/ Platform-as-a-Service (PaaS) delivery models as much as possible to achieve time to market and to minimize the operational overhead.
- Security: The system should provide the ability to protect data and information from unauthorized access while still providing authorized access to people and systems.
- HIPAA compliance: All components must ensure confidentiality of electronic-protected Health Information (ePHI) and conform to all HIPAA compliance protocols.
- Hierarchical Configuration Rules: The rules to trigger alerts should be hierarchically configurable to accommodate changes.
- Interoperability: Since the vendors follow their standards and formats, the data from different sources would require interoperability to convert the data into a more meaningful format.

Architecture

The architecture considers Amazon Web Services (AWS) as a cloud platform. It uses a combination of AWS Internet of Things (IoT) and time-series databases like InfluxDB. The choice for using an architectural component has been mentioned against the respective components.

Architecture diagram



Here is a breakdown of the architecture design components diagram:

AWS IoT Core

According to Amazon, "AWS IoT Core is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices." We also considered the Telegraf open-source server. We chose AWS IoT over Telegraf because:

- The application requires IoMT devices. AWS IoT provides built-in features for extracting data over Transport Layer Security (TLS) AWS IoT Core can also support billions of devices and trillions of messages.

- It enables compliance with HIPAA for protecting Protected Health Information (PHI) while achieving scalability.

Registry

The registry manages the devices by adding, updating and removing device information, making the control of permissions straightforward. It also categorizes and groups devices for ease of maintenance. Each device must have an X.509 certificate to communicate with AWS IoT.

IAM Policy

Tag-based resource-level permissions in the Identity Access Manager (IAM) policies are used to authorize AWS IoT API actions, giving better control over what resources can be accessed and modified. For example, an authorization could be provided to a specific IoT topic for a device.

Registering a device with AWS IoT

To register a device with AWS IoT, a certificate must be created and installed on the device. Each certificate is attached to an insurance policy, which helps in the authentication as the policy authorizes the device.

An option to enable auto-registration can be configured into the system so devices can register automatically. When the device tries to connect for the first time, it can send the client certificate signed by the Certificate Authority (CA) certificate.

Device communication

Devices send information to AWS IoT by publishing messages to MQTT topics. The connections to AWS IoT are made using TLS, so no client-side encryption keys are necessary for the initial TLS connection.

Amazon MSK

Amazon MSK is an open-source solution to process streaming data using Apache Kafka. According to Amazon, the metrics are ingested and processed later. Apache Kafka would help achieve decoupling by creating different components and would help achieve high scalability, as the processing would be done asynchronously.

Amazon MSK helps in achieving:

- High availability
- Scalability
- Loose coupling between components

AWS Lambda

The choice for AWS Lambda helps achieve scalability without bothering about the deployment or load management and achieving the elastic scalability at the same time.

- AWS Lambda would be used to ingest metrics to MSK from the devices
- Process: The metrics would then be processed, such as adding a device group name and details to the metrics. After transformation, it publishes the metrics to Amazon MSK.
- Publish: Takes processed data from the Amazon MSK and publishes it to InfluxDB.

Cross-cutting Concerns

Logging: Log activities by different components. It would help in debugging things in case of failure. For example, Lambda functions may write to logging if the data is incomplete or out of range.

Cloud watch: Cloud would monitor the system's overall health, such as detecting the number of JSON parse errors that occurred while processing requests. A number of failed job executions etc.

Audit: Audits help to ensure security measures are in place. It helps in detecting any drift, eg.: the device is still active even after revoking the CA certificate or the device certificate has expired, etc.

InfluxDB

InfluxDB, Prometheus and Graphite time-series databases were considered. The Graphite doesn't offer sharding and therefore lacks scalability.

Prometheus offers similar features compared to InfluxDB enterprise, but the language offered to query and create rules is very complex compared to InfluxDB. Flux provides an easy-to-use SQL-like query language for analyzing the database. Its enterprise edition offers features like shading, which helps to achieve scalability.

If cost is not the constraint, then InfluxDB is preferred over Prometheus. The rule configuration offered by Prometheus' alertmanager doesn't seem to provide a hierarchical configuration of rules, while InfluxDB's Kapacitator offers the same.

Kapacitator

InfluxDB's Kapacitator can configure hierarchical rule configuration and create alerts based on that hierarchy. It has an inhibit() function, which can be used to define rules to suppress the alerts within the hierarchy. Once an alert is created, the alert would be sent to AWS MSK.

AlertHandler Lambda

The Lambda function can detect alert messages sent by Kapacitator. Lambda would retrieve user profile information from the registry for the particular device and will send health-related data to the DynamoDB database.

DynamoDB

Amazon DynamoDB is a fully managed proprietary NoSQL database service providing high durability and availability. It can capture and track a user profile and process health information over some time.

For example, it can track and capture whether a user has been continuously going for walks or to the gym or following WHO standards for a period of time. It can interface with IoMT devices to monitor blood sugar levels or BP.

Rating Engine

The Rating engine is used to calculate the premium for a user based on inputs from IoMT devices in real-time to calculate a premium.

Chronograf

Chronograf is the user interface and dashboard for InfluxDB. It allows users to review stored data quickly and build alerts and queries, and it can also be used with Kapacitator to help configure various rules.

Insurance portal

Users would be able to create profiles, associate devices with the profile, get premium quotes and policies from this portal.

Conclusion

Insurance companies can take advantage of the IoMT devices to calculate the higher risk profiles and charge them accordingly. At the same time, they can categorize low-risk profiles and provide discounts to them.

The AWS IoT core and AWS components and InfluxDB could be a great combination to achieve a highly scalable system for dynamic premium calculations based on real-time factors.

About the Author

Ashutosh Gupta, Sr. Solution Architect at GlobalLogic, is a technology enthusiast with 20 years of experience. He has been associated with GlobalLogic India Pvt Limited for 13+ years and has significant experience in the design and architecture of high volume, load-balanced solutions in distributed applications. He is passionate about building solutions using cutting edge open source technologies.

GlobalLogic®

GlobalLogic is a leader in digital product engineering. We help our clients design and build innovative products, platforms, and digital experiences for the modern world. By integrating strategic design, complex engineering, and vertical industry expertise,— we help our clients imagine what's possible and accelerate their transition into tomorrow's digital businesses. Headquartered in Silicon Valley, GlobalLogic operates design studios and engineering centers around the world, extending our deep expertise to customers in the communications, automotive, healthcare, technology, media and entertainment, manufacturing, and semiconductor industries.



www.globallogic.com